



STOP PRESS

DRAGON

TELEPHONE COMPETITION

As mentioned in the editorial the 'mimic a telephone' competition proved to be very popular. After considerable help from a GPO expert we awarded two prizes, one to KEVIN PERRY (aged 11) of Quesport for a most realistic telephone sound, and one to IAN JONES (aged 8) from Sully for a very good imitation of the sound of a telephone ringing through the exchange.

Congratulations to you all and the two winners in particular.

1 REM TELEPHONE BY IAN JONES

20 FOR 1-1 TO 10

30 PLAY "TDR 035"

40 NEXT 1

50 FOR 1-1 TO 10

60 NEXT 1

70 FOR D=1 TO 10

75 NEXT D

80 FOR 1-1 TO 10

90 PLAY "TDR 035"

100 NEXT 1

110 FOR 1-1 TO 1000

120 NEXT 1

130 GOTO 20

1 REM TELEPHONE BY KEVIN PERRY

10 A1="0340040404"

20 B1=A1+A1+A1+A1+A1

30 C1=B1+"P3404"

40 PLAY B1

50 PLAY C1

60 PLAY B1

70 PLAY "P3404"

80 GOTO 40

EDITORIAL

Welcome once again to STOP PRESS. After four issues many of you know what to expect inside. Newcomers, however are invited to read on for new programs and articles and other program oriented material.

It is now one year since Dragon Data became an independent company. That year has seen the expansion of high-quality software, the marketing of disk-drives, and the continued development of new machines. Dragon 84 will be on general release shortly following its recent arrival on the US market. All that in one year! To celebrate the occasion Dragon Data have a special software offer - see elsewhere for details.

An apology is in order for the lack of time allowed for the last competition. But a big thank-you to many enthusiastic young readers (aged seven upwards) who sent entries for the telephone competition. Overwhelmed by such a response we called in a GPO engineer to help adjudicate! Congratulations to you all.

A number of letters are received after each issue from readers who have been unable to get a program to work. As is well recognised, type-setting and proof-reading of programs is very demanding. We make every effort to eradicate all mistakes but inevitably

human error creeps in. We do stress, however that there are a number of situations where it is easy for the reader to make mistakes. Be careful to distinguish between upper case I and the number one, and also between inverted commas surrounding a space and ones surrounding nothing!

The format of STOP PRESS is now under review and we hope plans for its continuation (and indeed expansion) will emerge in the near future. Readers' comments are welcomed - remember that STOP PRESS is produced for you!

This issue sees the first in a series of articles designed to introduce the reader to Dragon software, with a review of the editor-assembler cassette DREAM. Machine-Code Corner takes a look at producing an index with a more detailed look at the MC program next issue. For our young users the DRAW command is explored.

Plans are afoot to link Dragon owners via Micronet 800 to PRESTEL. A bank of 150 programs will be available for instant retrieval and readers are invited to submit programs for inclusion. As soon as possible.

Please address all correspondence to Miss Cathy Hyde, Dragon Data Ltd, Kentig Industrial Estate, Margam, Port Talbot, SA13 2PE.

MACHINE CODE CORNER



The animation of graphics using machine code can be spectacular but it is by no means the only area in which machine code can be usefully applied. An important use of computers is data retrieval, and the arrival of Dragon Disk Drives has enormously extended Dragon's capabilities in this area. Advanced data-retrieval systems are highly complex, but the basic ideas are within our reach.

The most important component is an index which can be searched efficiently, and efficiency means machine code. For the present, we shall use BASIC techniques to create an index, and a machine code routine to operate it. The following program allows you to enter words and page numbers (or other reference numbers) into an index.

```
10 CLEARMM
20 DIMA$(500),B$(500),A#(1000),B#(1000)
30 INPUT "READ FILE?";R:IF R<0 THEN 10
40 INPUT "FILE NAME?";F$
50 OPEN "A:";A:INPUT L$
60 FOR I=1 TO (INPUT L$;L$;NEXT;CLOSE:1
70 INPUT "ENTRY?";E:IF E="" THEN 30
80 INPUT "PAGE?";P
90 I=I+1
100 J=1
110 IF ASC(MID$(E,J,1)) < ASC(MID$(A,J,1)) THEN 200
120 IF ASC(MID$(E,J,1)) > ASC(MID$(A,J,1)) THEN 200
130 IF I=ASC(E) THEN 10
140 IF E=J THEN 10
150 IF E=J+1 THEN 10
160 B$(I)=E:IF E<>" " THEN 10
170 B$(I)=J:IF J<>" " THEN 10
180 INPUT "SAVE?";S:IF S<0 THEN 10
190 INPUT "FILE NAME?";F$
200 OPEN "A:";A:PRINT L$
210 FOR I=1 TO (PRINT L$;L$;NEXT;CLOSE:1
220 INPUT "PRINT?";P:IF P<0 THEN 10
230 FOR I=1 TO (PRINT L$;L$;NEXT;CLOSE:1
240 IF I=ASC(P) THEN 10
250 NEXT
260 INPUT "DELETE ENTRY?";D:IF D<0 THEN 10
270 INPUT "ENTRY TO DELETE?";X:IF X=1
280 IF X=ASC(E) THEN B$(I)=1:IF I=5 THEN PRINT "NOT FOUND":GOTO 10
290 GOTO 10
300 IF I=1 THEN 10
310 FOR I=1 TO (L$;L$;NEXT;CLOSE:1
320 IF I=1 THEN 10
330 IF I=1 THEN 10
340 IF I=1 THEN 10
350 IF I=1 THEN 10
360 IF I=1 THEN 10
370 IF I=1 THEN 10
380 IF I=1 THEN 10
390 IF I=1 THEN 10
400 IF I=1 THEN 10
410 IF I=1 THEN 10
420 IF I=1 THEN 10
430 IF I=1 THEN 10
440 IF I=1 THEN 10
450 IF I=1 THEN 10
460 IF I=1 THEN 10
470 IF I=1 THEN 10
480 IF I=1 THEN 10
490 IF I=1 THEN 10
500 IF I=1 THEN 10
```

Lines 30 to 40 input an index you have previously saved (first time around, answer N0 to the "READ FILE" prompt). Lines 70 to 170 add a new reference - if it matches a previous one, the page number will be added to the list. Lines 180 to 210 save the index on tape. Lines 220 to 250 list the index on the screen - if you have a printer, change the PRINT in line 220 to PRINT#2, for a hard copy. Lines 260 to 290 allow the deletion of an entry (just in case). In each section except the "ENTRY" section (adding a new reference) the first prompt requires a "YES" or "NO" reply.

If you have a Dragon Disk Drive you may like to substitute the following lines:

```
50 READMM
60 FOR I=1 TO (READMM;L$;NEXT
70 PRINTMM
80 FOR I=1 TO (PRINTMM;L$;NEXT
```

For small indexes, this method of storing the information is good enough, and the index can be interrogated by a BASIC program such as the following:

```
10 PMSO$="POLAR";CLEARMM
20 DIMA$(500),B$(500),A#(1000),B#(1000)
30 INPUT "FILE NAME?";F$
40 OPEN "A:";A:INPUT L$
50 FOR I=1 TO (INPUT L$;L$;NEXT;CLOSE:1
60 INPUT "WORD?";W:IF W="" THEN 10
70 IF W=ASC(W) THEN 10
80 IF W=ASC(W) THEN 10
90 IF W=ASC(W) THEN 10
100 IF W=ASC(W) THEN 10
110 IF W=ASC(W) THEN 10
120 IF W=ASC(W) THEN 10
130 IF W=ASC(W) THEN 10
140 IF W=ASC(W) THEN 10
150 IF W=ASC(W) THEN 10
160 IF W=ASC(W) THEN 10
170 IF W=ASC(W) THEN 10
180 IF W=ASC(W) THEN 10
190 IF W=ASC(W) THEN 10
200 IF W=ASC(W) THEN 10
210 IF W=ASC(W) THEN 10
220 IF W=ASC(W) THEN 10
230 IF W=ASC(W) THEN 10
240 IF W=ASC(W) THEN 10
250 IF W=ASC(W) THEN 10
260 IF W=ASC(W) THEN 10
270 IF W=ASC(W) THEN 10
280 IF W=ASC(W) THEN 10
290 IF W=ASC(W) THEN 10
300 IF W=ASC(W) THEN 10
310 IF W=ASC(W) THEN 10
320 IF W=ASC(W) THEN 10
330 IF W=ASC(W) THEN 10
340 IF W=ASC(W) THEN 10
350 IF W=ASC(W) THEN 10
360 IF W=ASC(W) THEN 10
370 IF W=ASC(W) THEN 10
380 IF W=ASC(W) THEN 10
390 IF W=ASC(W) THEN 10
400 IF W=ASC(W) THEN 10
410 IF W=ASC(W) THEN 10
420 IF W=ASC(W) THEN 10
430 IF W=ASC(W) THEN 10
440 IF W=ASC(W) THEN 10
450 IF W=ASC(W) THEN 10
460 IF W=ASC(W) THEN 10
470 IF W=ASC(W) THEN 10
480 IF W=ASC(W) THEN 10
490 IF W=ASC(W) THEN 10
500 IF W=ASC(W) THEN 10
```

After you have given the name of the data file which contains the index, the index will be loaded. The prompt WORD? requires a reference word, which the computer will "look up" for you. Note that lines 70 to 90 perform a simple "search pattern" to avoid searching the whole index. Disk owners may substitute:

```
40 READMM
50 FOR I=1 TO (READMM;L$;NEXT
```

For large indexes, these methods are rather slow, both in creating indexes and in interrogating them. Machine code would help in both of these exercises, but we shall concentrate here on the interrogation side, since an index is usually created only once, but may be interrogated thousands of times.

The first job is to create a file of data which is easy to search using a machine code program. There are many ways of doing this, and the most common methods use the ASCII codes to

represent the letters, "setting" the most significant bit of the last letter of each word. In other words, 128 is added to the code for the last letter to indicate "end of word". We shall use this method, inserting the reference pages after each word, and ending with a zero for "end of reference". Finally, two zeroes in succession indicate "end of file".

The following BASIC program will convert your data file to the required "binary" file:

```
10 POKE1,CLEAR:10000:A=10000:B=A
20 INPUT"FILENAME";F$
30 OPEN"1",F$:IN=INPUT#1,F
40 FOR I=1 TO IN:INPUT 1,A1,B1
50 FOR J=1 TO LEN A1
60 POKE1,ASC(MID A1,J,1)*B+1:INEXT J
70 POKE1,128+POKEB-1
80 J=1:B=2
90 IF LEN B < 8 THEN B=
100 IF MID B,8,8="" THEN B=
110 B=B+1:GOTO 60
120 POKE1,VAL(MID B,J,8)-B+1
130 J=J+8:B=J+2:GOTO 60
140 POKE1,VAL(MID B,J,8)-B
150 POKE1,128+B-1:B=1536+320 THEN PRINT"FILE TOO LARGE":STOP
160 INEXT J:POKE1,CLOSE-1
170 INPUT"FILENAME";F$
180 G$=VEN$+A,B,1
```

The disk version will have:

```
30 FREEDIM A
40 FOR I=1 TO IN:FREE DIM A1,B1
50 SAVE M1,A,B,1
```

If you want to see a list of words already stored in your Dragon, have a look at ROM addresses, starting at 32819.

This final BASIC program POKEs in 66 bytes of machine code, loads the binary file containing your index, then uses the machine code to interrogate the index.

```
10 DATA 95,15,AE,24,55,32,F7,7F,7F,5E,37,78,55
20 DATA 2AC,2F,24,8F,7F,7D,AA,AB,8B,32,3A,AF,8A
30 DATA 25,19,22,14,5A,37,32,8F,1A,AB,7F,6E,22,19
40 DATA 1,8B,25,8B,22,8B,5A,25,7F,25,8F,37,7D,7E,7F
50 DATA 7F,AB,8B,25,7C,AB,AB,25,32,3B,AB,8B,AF,8B,25
60 DATA 8B,22,8B,5A,25,8B,AB,AB,AF,32,25,7A,3B
70 CLEAR:1,CLEAR:10000:X=""
80 FOR I=32819 TO 32879:READ X:POKE1,VAL("0"+X):NEXT I
90 INPUT"FILENAME";F$
100 CLOAD M1
110 INPUT"WORD";X:X=ASC(X)
120 IF X=65535 THEN B=1:IN=1
130 EX=0:NN=
140 A=0
150 B=POKE1,IFX=0 THEN 170
```

```
160 POKE1,A-A+1
170 X=POKE1,IFX=0 THEN 160
180 PRINT:GOTO 160
190 PRINT:PRINT"NOT FOUND":GOTO 160
```

The disk version will have:

```
100 LOAD M1+".BIN"
```

The detail of the machine code will be investigated in a future article. Its purpose is to search the index for the word input, and to "POKE" the references into the memories starting at address 1000. The address following the final reference is then cleared to zero. If the word is not found, memory 1000 is set to zero. These results are then interpreted by lines 140 to 150 of the BASIC program. Note that the use of one byte per reference restricts us to the range 1 to 355, but this can be extended by using 2-byte references.

The response of this program is almost instantaneous, even with several thousand words in the index, whereas the BASIC version gets unacceptably slow after the first few hundred.

DRAGON PUZZLE



10 CLS:PRINT"DRAGON PUZZLE"	
20 PRINT@15,"....."	drop, burning.
30 PRINT@20,"....."	revolt.
40 PRINT@27,"....."	assault-pointed missile.
50 PRINT@32,"....."	game played on squares.
60 PRINT@38,"....."	search.
70 PRINT@45,"....."	mad.
80 PRINT@50,"....."	Greek god of the sea.
90 PRINT@55,"....."	bodies moving through space.
10 PRINT@60,"....."	

Clue

```
100 FOR I=1 TO 100:PRINT I:P=
P=POKE1-1000:PRINT@I,CHR$(P+32):NEXT I
```

ISSUE NUMBER 5

Many of our readers noticed that the latest issue of Stop Press was incorrectly numbered. It was, however, issue number 4 and this is issue number 5. We apologise for the confusion this may have caused.



YOUNG USER PAGES

DOODLING

Are you good at doodling? It's fascinating watching a picture build up isn't it? Perhaps you prefer to know exactly what you want before you start, planning the whole thing before you put pen to paper. Whatever you prefer, you can build up pictures on your Dragon as easily as doodling.

We've already looked at the **CIRCLE** command and we know that some of you are experts at using it because we have seen the excellent entries to the Dragon Logo competition. This time let's look at the **DRAW** command and doodle with that. It's useful to start with a tiny program which will enable you to type in **DRAW** commands and see the results. Here it is.

```
10 CLS:POLARPP:MODE3:POL3
20 PRINT "ENTER INSTRUCTION ":
30 LINE INPUT D1
40 SCREEN1:DRAW D1
50 IF INKEY="" THEN GOTO 20
```

The first line sets up the high resolution graphics screen and clears it. It also clears the Text screen. The next two lines tell you what to input and take in the instructions for your drawing. **LINEINPUT** is used because it will accept commas. Line 50 reveals the screen and does the **DRAWING**. The last line holds the screen in view until you press a key. Then you are ready for a new input.

Type the program and then **RUN** it. Read on for an explanation of its use.

It's as though your Dragon were holding a pen at the back of the screen. When you start it is in the middle of the screen and you can tell your Dragon how to move the pen.

The easiest instructions you can give are to draw up, down, left and right. The first letters of the direction are used. So **U20** means up 20 pixels (the points on the screen). **L35** means left 35 pixels. **INPUT U20** and you will see a line on the screen. Now press any key to return to the print screen. This time **INPUT L35**. You can put several instructions in at once separating them with a semi-colon (;). The semicolon isn't always necessary but if in doubt put it in. Now try **D5:R15;U35:L**.

The next set of instructions are not so easy to remember. They use the letters **E,F,D** and **H** and I think a diagram is the best way of explaining them.

Try **E50;F10** and **G20H4**.

Well that's fine for doodling without taking your pen off the paper. If we want to lift the pen up and make it move without drawing we simply put



a **B** (for blank) in front. So **B50:L5** will take the pen down 50 pixels without making a mark and then draw leftwards for five pixels. We can also make the pen draw in one direction and go back to its starting position before the next drawing. Try this one: **L10;R10;R10;R10;B10**, an impression of a pigeon standing on one leg! (The **R** stands for 'no up date of pen position').

Of course sometimes these eight directions aren't enough and we want to draw a line at a different angle. Well if we want to move the 'pen' to the point 123 pixels from the left hand side of the screen and 47 pixels down we just say **R123;47** - try it. Put **B** in front if you want to move the pen without drawing.

You can change the pen colour too. **C7** changes the pen to magenta. **C5** draws in the background colour and is useful for rubbing out. First try **C70** 50 then try **C550**. You had better type in **C5** to get back to the orange pen.

Now you are ready to doodle using the program. Use **BREAK** and **RUN** when you want to start afresh with a clean screen.

You have learned how to write strings which form instructions to the **DRAW** command. You may like to use such a string in a program. The short strings can be added together to make one long one. I have got a string which draws a wiggly worm. I'll call it **W5**. We'll put it into our program as line 20. Just type in:

```
20 W5="--F50;F50;F50;F50;L20;H10;B50;R50;R50"
```

Now we can explore some more commands which you will find useful. We have a string called **W5**. We can command it to be drawn by entering the string **XW5**; you see you have drawn the worm. (The semi-colon is essential here.) Now try **XW5;R100W5**. Now you have the worm twice. You can change the size of the worm with another instruction.

S50W5; draws a large worm.

S10W5; draws a small one.

If you want the worm to stand on its head try **A10W5**; if you try **A20W5**; you will find the worm has turned! Experiment by putting other numbers after **S** or **A**.

Finally by using the background colour to wipe out our worm and moving it to the right we can make it creep across the screen as you press the space bar. We need to rewrite lines 30, 40 and 50, and modify line 58. This is our new program.

```
10 CLS:POLARPP:MODE3:POL3
20 W5="--F50;F50;F50;F50;L20;H10;B50;R50;R50"
30 DRAW "BLNK"
40 D1="--C5;W5;C5;L50;"
50 SCREEN1:DRAW D1
60 IF INKEY="" THEN GOTO 40
```

You will find all the details about the **DRAW** command in the programming book which came with your Dragon.

PLAY HOUSE

This program is for the very young! If you have a child in the family who is just beginning to recognize the first letters of words then why not try it? After typing the program in and running, the keys H (for House), W (for Window), D (for Door) C (for Chimney), P (for Path, T (for Tree) and A (for Again) are the only keys that are active and pressing each key draws the object it stands for. However, the position it draws it in will only be the correct one if the sequence H D W W W W C P T is used. This allows for some fun if you press C instead of W and have a house with a chimney where a window should be!

As you can see, the program is quite short and uses the `DRAW` command. You might think that is a little strange when it comes to windows (where we could have used the `LINE` commands).

However, use of DPAW protects the program from crashing if the picture goes outside the screen.

The same idea could be used in many different situations using other words - all that is required is the same program structure but with lines 100 to 160 replaced by other appropriate sub-routines to draw the objects.

[illegible]

Note that in line 50, before the JOYSTK function is called, there is an EXEC32780. This is necessary because JOYSTK(0) is not always executed at this point. The joystick status are only updated when JOYSTK(0) is executed. When other values (e.g. JOYSTK(2)) are executed, the number returned is the one which was sampled last time JOYSTK(0) or EXEC32780 was performed.

PAGE

Here is a joystick game for two players. After the race track is set up, and you have said how many laps you want to race, the first to pass "line" is away in car number 5. Each turn, the cars travel in a straight line for a distance which depends on how far the joystick is pushed over. If this results in a collision with the other car, or with the side of the track, you go back to where you started that turn. The race takes place in an anti-clockwise direction.

[illegible]

LETTERS FROM READERS

Thank you to all readers who wrote to us with comments and programs. Four readers' programs are included with slight modifications to economize on space.

Patlerna was the theme of two programs written by P.C. Asbury-Smith ('Random Fantasy') and John Oliver ('Tunnel'). Mr Ian Robertson sent us his barograph program which accepts monthly frequency data and constructs a barchart. The statement 50 READ MN) can be replaced by 50 INPUT MN) and the demo data removed by deleting line 220.

A program to convert joystick readings to 'keypad' numbers was sent by J. Wrennall. The crux of the program lies in a clever use of a user-defined function in line 20.

We are grateful to N. Atkinson for drawing attention to an error in the program ISLANDS in the last issue. Lines 530 and 540 need to be amended to the following to eradicate an occasional problem that occurred with the original version.

```
530 IF D1 < -1 THEN D11 = -1 + ABS(D1)
540 IF D1 > 254 THEN D11 = 254 - D1 ELSE D11 = D1
+ SGN(D1)
```

Note too that a number of lines were missed out in the introduction to the program which showed how it could be converted to use a joystick. Simply delete lines 510 and 520 and replace lines 530 and 540 with the given lines.

```
10 Z = 0: REM IT'S VALUES FROM STICK: J. WRENALL
20 DEF FN D1 = 0 + SGN(JOYSTICK(1)) * ABS(JOYSTICK(1) - 254)
+ SGN(JOYSTICK(2)) * FN D1
30 CLS: PRINT@80, "STICK STORED"
40 PRINT@80, "VALUE VALUE"
50 PRINT@80, "1 2 3"
60 PRINT@80, "NOTE THE CHANGE IN STICK VALUE AS YOU MOVE THE JOYSTICK AROUND"
70 PRINT@80, "AND THE STORING OF THE VALUE AS YOU PRESS THE BUTTON"
80 PRINT@80, "CHANGE THE 'Z' VALUE IN LINE 10 TO ALTER STICK SENSITIVITY"
90 PRINT@80, "press break to quit"
100 PRINT@160, FN D1
110 FOR D = 1 TO 100: NEXT D
120 P = PEEK@256
130 IF P = 128 OR P = 254 THEN A = FN D1: SOUND(1)
140 GOTO 100
```

```
10 REM RANDOM FANTASY P.C. ASBURY-SMITH 1982
20 CLS
30 X = 31 * Y = 10 * INT(RND(1))
40 FOR N = 1 TO 250
50 D = 2 * RND(1) * X + D: E = 2 * RND(1) * Y + 1
60 IF X = 0 THEN X = 31 ELSE IF X = 31 THEN X = 0
70 IF Y = 0 THEN Y = 31 ELSE IF Y = 31 THEN Y = 0
80 SETX, X: RND(1): NEXT N
90 GOTO 0
```



```
10 REM DRAGON BAROGRAPH IAN ROBERTSON JUNE 1982
20 DIM M(12)
30 FOR N = 1 TO 12
40 PRINT "ENTER DATA FOR EACH MONTH"
50 PRINT@80, "MONTH": N
60 READ MIN: MIN = MIN * 10
70 IF MIN > 101 THEN MIN = 101
80 CLS: NEXT
90 FOR M = 1 TO 12: READ MIN: NEXT
100 PRINT@80, "TO SEE GRAPH PRESS speaker"
110 PRINT@80, STRING$(12, " ")
120 PRINT@80, "speaker AGAIN FOR NEW GRAPH"
130 SCREEN 1
140 IF INKEY = " " THEN 140
150 PMODEL, 1: POL: SCREEN 1
160 N = 12: FOR M = 1 TO 12: DRAW "BM" + STRING$(12, " ")
+ AINT: M = N + 20: NEXT
170 C = 2.5 * 7: COLOR C: FOR M = 1 TO 12: COLOR C
180 LINE, 100 - X + 20: MM: M: SET B: COLOR
190 LINE, 100 - X + 20: MM: M: SET B
200 X = X + 20: C = C + 1: IF C > 4 THEN C = 2
210 NEXT
220 DATA 10,15,10,10,10,15,10,10,10,10,10,10
230 DATA: BARO(1), BARO(2), BARO(3), BARO(4), BARO(5), BARO(6), BARO(7), BARO(8), BARO(9), BARO(10), BARO(11), BARO(12)
240 DATA: BARO(1), BARO(2), BARO(3), BARO(4), BARO(5), BARO(6), BARO(7), BARO(8), BARO(9), BARO(10), BARO(11), BARO(12)
250 IF INKEY = ">" THEN 250 ELSE RUN
```

```
10 CLS: REM TUNNEL JOHN OLIVER
20 INPUT "THE VALUE OF STEP": Z
30 PMODEL, 1: SCREEN 1: POL
40 A = RND(1) * 8 + RND(1) * 8: B = RND(1) * 8 + RND(1) * 8
50 A = A + 2.5 * B + 2.5 * B + 2.5 * B + 2
60 GOSUB 200: IF P = 1 THEN 60
70 LINE(1, 0) - (X, Y): SET B
80 GOTO 0
90 A = A + 2.5 * B + 2.5 * B + 2.5 * B + 2
100 GOSUB 200: IF P = 1 THEN 100
110 LINE(1, 0) - (X, Y): SET B
120 GOTO 0
200 P = 0: ON A + 25: ON B + 0: ON B + 25: ON X + 0: ON X + 25: ON Y + 0: ON Y + 25: RETURN
```

Pleasant DREAMs



Each issue of Stop Press has included machine code programs both in their mnemonic form and in their binary form. Developing such programs needs the facility of an assembler program. DREAM is such a program available from Dragon Data Ltd. The cassette-based program comes with a comprehensive booklet explaining its ability to edit text (of any sort) and to assemble program text files into machine code. Together with DREAMBUG (also on cassette) or combined, in a cartridge (ALLDREAM) DREAM is ideal for implementing machine code programs. What follows is designed to provide you with an introduction to DREAM and present a working session using it to assemble a program. To do this all key-strokes are shown together with an indication of their results. Where necessary, keys such as ENTER and BREAK are shown in square brackets. In addition SHIFT together with SPACEBAR (used for tabulating between fields) is presented by [TAB].

To illustrate the use of DREAM we will enter the text version of a program we included in Stop Press 1. First load DREAM and press 'N' for a new text.

```
CLEAR 200,20000 (ENTER)
OLDORIG DREAM (ENTER)
N
```

Now key in each line of the text.

```
(TAB)      LDA      (TAB)  OFF      (ENTER)
(TAB)      LDX      (TAB)  #400     (ENTER)
LOOP      (TAB)  STA      (TAB)  X      (ENTER)
```

(Oops - missed the '+' out)

```
+ (TAB) (TAB)  S+ (ENTER)
```

(Move up to previous line, tab to 'X' and add '+')

```
(TAB)      CMPX      (TAB)  #000      (TAB)
(TAB)      BNE      (TAB)  LOOP      (ENTER)
(TAB)      RTS      (ENTER)
```

We now have the text of the program as in the newsletter - complete with mistake - it should have been CMPX #000, this # denoting an immediate address. To correct this either use the arrow keys to tab to the appropriate place or use the EDITOR as follows.

```
(BREAK) H (ENTER)      returns cursor to home position
(BREAK) C/0: / #0: (ENTER)
```

The editor scans down the text file to find the first occurrence of #0 and replaces it with #00.

There are a few points to note here. First, you must identify uniquely, that part of the text you wish to alter - it would be no use using the command C/0:0/ from the home position as it would alter the first 0 sign it came to. Secondly, note that C/0:0/ would delete the first occurrence of #0.

Finally, in some situations we need to change all occurrences of a pattern of text. To do this use an 'A' at the end of the command. This can be particularly useful when a complicated piece of text occurs frequently - substitute say ZZZ for the piece and after executing the HOME command use the command C/ZZZ/whatever it should be (A).

DREAM's editor facilities include many other features. Although our text is ready to assemble we will demonstrate some more commands.

```
(BREAK) H (LOOP)
(BREAK) F/LOOP (ENTER)
```

(Finds the first occurrence of 'LOOP'. If you wanted to find all occurrences repeat the last command using

```
(SHIFT) (J)
(BREAK) C/ (ENTER)
```

(Deletes two lines starting in the current position.)

```
(BREAK) H (ENTER)
↑ ↑ (BREAK) C (ENTER)
```

(Prepares for insertion of two lines in the position of the cursor.)

```
LOOP      (TAB)  STA      (TAB)  X+      (ENTER)
(TAB)      CMPX      (TAB)  #000      (ENTER)
(BREAK) H (ENTER)
```

(Text now restored to previous state.)

Continued on next page

Pleasant DREAMs (continued)

To assemble the program type **[BREAK] A [ENTER]**. The resulting screen has as its first line **4E21 B07FFF LDA 07FFF**. This shows that the bytes of code **B0, 7F, FF** have been assembled starting at memory location **4E21** or decimal **20001** which is the default setting used by DREAM. If you wish to place the code elsewhere then use an **ORG** instruction as the first line of the test program.

As you should see there are no errors. What happens if there are?

```
[BREAK] Q [ENTER] (returns to text file)
[BREAK] H [ENTER]
[BREAK] C/STA/STZ[ENTER] (changes STA to STZ which
                           is wrong.)
[BREAK] A [ENTER]
```

Now the screen shows an error on the incorrect line. Type **[ENTER]** again to continue assembly. Now to restore the text, type: **[BREAK] Q [ENTER] [BREAK] H [ENTER]**
[BREAK] C/STZ/STA/[ENTER]

Re-assemble using **[BREAK] A [ENTER] [BREAK] Q [ENTER]**.

It is important to save the test file as soon as it has assembled without errors as running the machine code program could corrupt any of the contents of RAM. If you wish to manipulate a tape, exit to BASIC using **[BREAK] Q [ENTER]** from text mode and you are free to control the tape recorder.

It is a good idea to keep a separate tape for DREAM test files as the **SKIP** command does not function with such files. (This is so that test files may be merged from tapes.) Positioning the tape can be achieved by using **MOTOR ON, AUDIO ON** and **PLAY**. Then listen for the sound (ack) indicating a clear space on the tape. Then type **EXEC 27778 [ENTER] Y** (to return to the old text) followed by **[BREAK] SCREEN [ENTER]**.

Note that the space between **S** and the title is necessary. Each line of text disappears as it is loaded until finally the HOME position is repeated.

Finally we will show how rearing of files can be achieved by adding an **ORG** statement to the recorded program.

```
[BREAK] Q [ENTER]
EXEC 27778 [ENTER] (TAP) ORG (TAP) HPM [ENTER]
```

Prepare the recorder to play back the stored program and type **[BREAK] L SCREEN [ENTER]**. DREAM will search through the tape until **SCREEN** is found and load it after the current line.



A few readers have experienced difficulties with the saving and loading of DREAM text files. These difficulties can be avoided as follows:

If the Header information on your tape is being corrupted then **POKE 6HT46B,255**.

If the Test File will not load correctly then **POKE 6HT46B,128**. If this is successful, then gradually decrease the **POKE** value from 128 to a minimum of 1, but **DO NOT USE POKE 6HT46B,0** as this will produce an extremely long Header. If unsuccessful, gradually increase this value from 128 to 255.

DREAM has many other facilities (for example test programs may be sent to a printer using the editor command **P**) but we have used those which are most useful for our simple machine-code programs. If you wish to implement the material in MCC why not try DREAM?



WIN A DRAGON 64

We are looking for the Dragon User of 1983. If you have an unusual application story for your Dragon 32 whether at home, at school or at work then we want to hear from you.

Send a report outlining in less than 100 words on how you use your Dragon computer to:

Cathy Hyde,
'Dragon User of 1983',
Dragon Data Ltd,
Kensington Industrial Estate,
Margam,
Port Talbot,
West Glamorgan.

Entries must be received by November 30th. The winner will receive a new Dragon 64 computer.

The judges' decision will be final. The winning application story will appear in a future issue.